Tackling Bufferbloat in Capacity-limited Networks

Chamil Kulatunga, Nicolas Kuhn, Gorry Fairhurst, David Ros Sanchez

Tackling Bufferbloat in Capacity-limited Networks

Chamil Kulatunga†, Nicolas Kuhn‡, Gorry Fairhurst†, David Ros⋆
†School of Engineering, University of Aberdeen, Aberdeen, UK. {chamil,gorry}@erg.abdn.ac.uk
‡IMT Télécom Bretagne, IRISA, France. nicolas.kuhn@telecom-bretagne.eu
⋆Simula Research Laboratory, Fornebu, Norway. dros@simula.no

***Abstract*—Over-provisioned network buffers, often at the Internet edge, induce large queuing delay and high latency; this issue is known as Bufferbloat. In response to this, a set of recently proposed Active Queue Management (AQM) algorithms attempt to reduce standing queues, while maintaining the bottleneck utilisation at an acceptable level. This paper assesses the performance of two AQM schemes (CoDel and FQ-CoDel) over capacity-limited networks with large Round-Trip Time (RTT). In such settings, these AQM schemes have difficulty controlling the buffering level, resulting in both momentarily high queuing delay and low bottleneck utilisation, even if the methods are claimed to be insensitive to link rates and round-trip delays. We explore this issue and show that it is possible to adapt the parameterisation of CoDel and FQ-CoDel to offer a higher bottleneck utilisation while maintaining a low queuing delay. We present experiments over an emulated test bed and a satellite network to confirm that our new parameterisation improves the download time of moderate-size files and reduces the latency for capacity-limited and large-RTT networks.**

***Keywords*-Bufferbloat, Rural Broadband, AQM, CoDel.**

I. Introduction

Oversized buffers along Internet paths may lead to large standing queues and high queuing delay, this issue is known as *Bufferbloat* [1]. Active Queue Management (AQM) proactively drop packets before router buffer space is exhausted. This signals incipient congestion to endpoints and avoids persistently large queues. AQM aims at reducing queuing latency and is one piece of the solution to Bufferbloat.

Random Early Detection (RED) [2] is an AQM proposed two decades ago, but has not been widely enabled in Internet routers due to the need to tune its parameters depending on the actual traffic and network conditions. Newly proposed AQM algorithms, such as Controlled Delay (CoDel) [3], Proportional Integral controller Enhanced (PIE) [4] and Flow-Queuing CoDel (FQ-CoDel) [5]), claim to address this difficulty of configuring AQM parameters.

In a rural broadband network, a service is often characterised by much less capacity than in other contexts. The greater cable distance to many rural locations is one key factor that can limit offered capacity, but the type of installed access technology and the lower population density can also impact available capacity. In the UK, 21% of rural areas are currently unable to access to a DSL downlink speed of 2 Mbps [6]. Despite introducing a larger RTT (e.g., 500 ms or more) satellite technology may be deployed in such cases to realise a commercially viable broadband service [7].

In [3], the authors say that CoDel "controls delay, while insensitive to round-trip delays, link rates, and traffic loads". The claim that CoDel can self-tune its dropping policy suggests that no knobs needs to be tuned. However its behaviour has been shown to depend on the congestion level [8] and the RTT of the path [9]. The benefits of introducing AQM mechanisms in capacity-limited and large RTT networks are not yet well understood.

In this paper, we evaluate the potential of deploying AQM in such networks to tackle Bufferbloat. We propose to tune the parameterisation of CoDel and FQ-CoDel to both limit the queuing delay and optimise use of resources for a large-RTT and capacity-limited network. Evaluation in an emulated test bed shows that our parameterisation reduces the transfer time for medium-sized files. This analysis is supported by tests using a broadband satellite access network.

The remainder of this paper is structured as follows. Section II details the CoDel and FQ-CoDel algorithms. Section III illustrates how low-capacity, high-RTT paths are challenging for these self-tuning AQM schemes. Based on ns-2 simulations, Section IV develops our updated parameterisation of CoDel. Section IV also evaluates the impact of various base RTTs on the performance of CoDel and FQ-CoDel. Experimental results, including from an actual satellite broadband link, are shown in Section V. Section VI concludes the paper.

II. Delay-based AQM schemes

Delay-based AQM algorithms typically use a target delay, τ, determining the allowable standing queuing delay, and an update interval, λ, determining the frequency at which the dropping policy is updated. PIE has already been shown to need an updated parameterisation for specific scenarios (such as data-centers [10] or cable modems [11]). CoDel has been shown to not always control the queuing delay with a limited impact on the bottleneck utilization [8], [9], and it was uncertain whether CoDel could be tuned for objectives and network conditions other than the ones for which it has been designed: this led to our interest in examining CoDel and not PIE. We also wanted to evaluate the performance of a hybrid scheduling/AQM scheme, which is possible with FQ-CoDel, whereas there is no reference algorithm for FQ-PIE. Therefore, we will focus on CoDel and FQ-CoDel, for which $\tau = 5\,\text{ms}$ and $\lambda = 100\,\text{ms}$ have been suggested as default values [3], [5].

A. CoDel

CoDel [3] tracks enq_i, the *enque* time of each packet p_i. At the *deque* time, deq_i, CoDel computes the queuing delay of

each dequeued packet: $\delta_i = deq_i - enq_i$. CoDel has two states, *dropping* and *non-dropping*, and starts on the latter state. We denote by μ the interval after which CoDel may change its state and drop one packet. The initial value of μ is λ. We denote by n_{drop} the number of consecutive drops, initialized as $n_{drop} = 1$. Between $t_1 = t$ and $t_2 = t + \mu$, if there is a packet i such that $deq_i \in [t_1, t_2]$ and $\delta_i > \tau$, then: (1) CoDel enters the dropping state, (2) the next packet to be dequeued is dropped, (3) n_{drop}+=1 and (4) μ is set to $\mu/\sqrt{n_{drop}}$; else: (1) CoDel enters the non-dropping state, (2) there is no packet drop, (3) n_{drop} is reset to 1 and (4) μ is reset to λ.

B. FQ-CoDel

FQ-CoDel [5] maintains one sub-queue per flow. There is one instance of CoDel per sub-queue. FQ-CoDel features priority-queuing by giving priority to new incoming flows and fair-queuing. When a packet arrives, if there is already another packet belonging to this flow in an existing sub-queue, the incoming packet is added to the latter, else a new sub-queue is instantiated and the incoming packet added to it. FQ-CoDel maintains two lists of sub-queues, *new* and *old*. When looking for a packet to dequeue, FQ-CoDel starts by looping over the *new* list. If the selected sub-queue has already dequeued a quantum of bytes (default 1514 B), this sub-queue is put at the end of the *old* list. If the selected sub-queue in the *new* list has dequeued less than a quantum of bytes, the sub-queue is selected. If there are no sub-queues in the *new* list where less than a quantum of bytes has been dequeued, FQ-CoDel loops over the *old* list in the same manner. Once a sub-queue has been selected, the dropping policy of CoDel is applied to the sub-queue: a packet is dequeued from it if the algorithm of CoDel does not drop this packet.

III. AQM schemes in capacity-limited networks

A. Topology and traffic

The topology presented in Fig. 1 was used with experiments run with Linux kernel 3.14.4 and a network emulated by `netem`. Throughout the paper, we consider that there is a sender/receiver pair for each flow (2 flows in this section), all of them sharing the same bottleneck. All flows use TCP NewReno with SACK and an initial window of 3 segments.

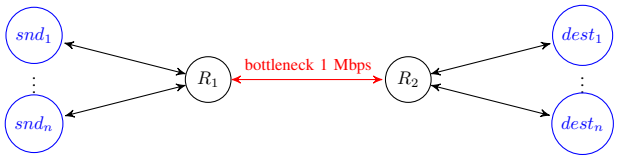


Figure 1: Dumb-bell simulation topology.

The symmetrical bottleneck between R_1 and R_2 has 1 Mbps capacity and a range of RTT values were tested. The bottleneck buffer size is 25 packets of size 1500 B (i.e., a 300 ms worst-case queuing delay). Other links have a 1.25 ms one-way delay and 100 Mbps capacity with a DropTail buffer of 300 packets. AQM schemes are applied at node R_1 but not at R_2. In this section, the default parameterisation of CoDel and FQ-CoDel is used (τ = 5 ms and λ = 100 ms). 300 s, was replicated ten times and the following plots represent average values and 95% confidence intervals.

B. Issues with low-capacity links

We consider one bulk transfer in parallel with repeated downloads of a 1.7 MB file (inter-file intervals are exponentially distributed with mean 9.5 s). In Fig. 2, we plot the throughput of the bulk transfer (averaged every second) and the average File Completion Time (FCT) for the repeated downloads. Queueing delays (not shown) were as high as 300 ms with DropTail, whereas with CoDel and FQ-CoDel they tended to stay below ≈ 30 ms (i.e., at most a couple of packets in the queue).

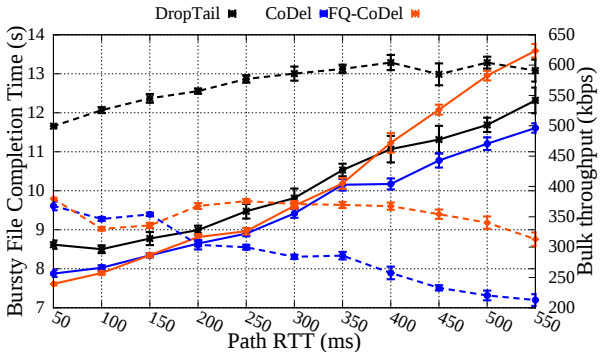


Figure 2: Performance with different queue management methods, for various path RTTs (solid lines: FCT for a repeated download; dashed lines: bulk transfer throughput).

These results illustrate the main problem with the considered AQMs when bottleneck capacity is low. AQM does maintain a much lower queuing delay, which should help with achieving a lower FCT. However, the net effect of AQM is a large drop in bulk throughput, for both short and long RTTs, for a relatively modest gain in FCT (≈ 10% at best, for medium-sized files).

Such degradation of throughput can be intuitively explained as follows. With a 1 Mbps link, it takes 12 ms to transmit one 1500-B packet. An AQM algorithm with a 5 ms target delay over a 1 Mbps link is thus attempting to maintain a queue of less than one packet, and an often empty queue translates into lower throughput and low utilisation.

IV. Tuning CoDel to capacity-limited networks

Our goal can be stated as: *we seek to find a combination of τ and λ for both CoDel and FQ-CoDel that, for capacity-limited networks, both enables a high bottleneck utilisation (around 90% or higher) and limits the queuing delay*. We consider both τ and λ because the resulting queuing delay and bottleneck utilisation depend on their interaction: drops are only applied if the queuing delay grow above τ and only every $\mu_{n_{drop}}$ in order to give end-points time to react to a drop.

Figure 2 shows that throughput degradation with FQ-CoDel is less sensitive to the RTT, compared with CoDel; more

generally, the flow-queuing mechanism may result in different dynamics for the individual flows. However, if multiple flows coexist in a single queue (e.g., when flows are aggregated in a VPN), FQ-CoDel would behave similarly as CoDel. Therefore we will consider a parameterisation for CoDel, and verify whether it suits FQ-CoDel too. In this section, we use the ns-2 simulator to easily cover a wide range of values of τ and λ. The validity of our parameterisation is verified with emulations in Section V.

A. Topology and traffic

The topology shown in Fig. 1 was simulated with ns-2. The symmetrical bottleneck has 1 Mbps capacity and a 300 ms RTT; the bottleneck buffer size is set to 25 packets. Other links have a 1.25 ms one-way delay and 100 Mbps capacity with a DropTail queue of 300 packets. On the bottleneck link, CoDel and FQ-CoDel are used in node R_1 but not in node R_2.

The following traffic was considered: (1) repeated downloads of a file of size 1.7 MB, with an inter-file interval exponentially distributed with mean 9.5 s; (2) TCP bulk transfer, lasting for the entire length of the simulation; (3) "thin" TCP flows with a constant inter-packet interval of 300 ms and a packet size of 150 B; these may represent gaming traffic. Both (1) and (2) use a packet size of 1500 B. TCP characteristics are the same as in Section III-A. Two traffic mixes were simulated: *low* load (one flow for each of flow types (1), (2) and (3)) and *high* load (two flows for each of (1), (2) and (3)). The flows start randomly within the first second of the simulation. The simulations last 300 s and were run 100 times, each with a different seed.

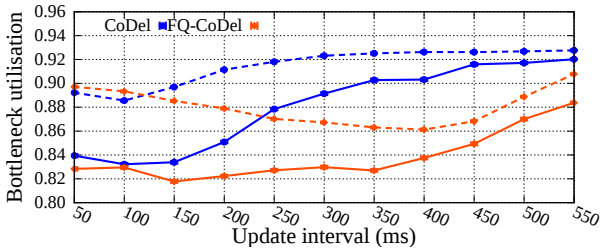
B. Tuning λ to maximise the goodput

We first want to find a λ so that $\approx 90\,\%$ of the link capacity is exploited while limiting the impact on the queuing delay.

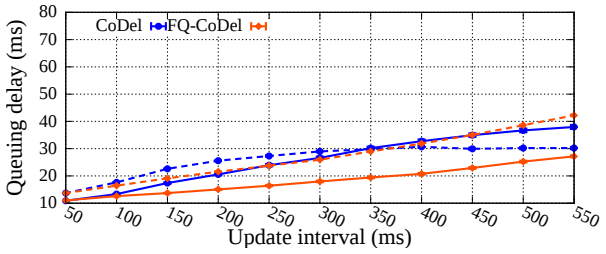
The authors of [12] explain that λ should be set to the largest RTT of the flows sharing the bottleneck. Such setting is said to allow CoDel to keep control of the queuing delay. However, in many deployment cases, the person configuring the router will not be aware of the path RTT. Moreover, in a low-capacity scenario, changing λ alone may not be enough to obtain both a good bottleneck utilisation and control of the delay. When λ increases, CoDel reduces the frequency at which it updates the dropping policy. A larger λ induces less packet drops, resulting in a higher bottleneck utilisation at the expense of higher queuing delay. This can be seen in Fig. 3. With CoDel, increasing λ improves utilisation, with λ = RTT = 300 ms bringing utilisation reasonably close to 90% for both traffic profiles. But, for all the values of λ tested (which include the default one of 100 ms), delays are always $> \tau$.

C. Tuning τ to maximise the goodput

We next turn to finding a value of τ so that at least 90 % of the available capacity is exploited while limiting the impact on the queuing delay. Based on the results in §IV-B, λ is set to 300 ms for both AQMs. Figure 4 presents the same performance metrics of Fig. 3 for various τ. The choice $\tau =$ 65 ms yields a reasonable compromise between queuing delay (< 50 ms) and utilisation (slightly above 90%).

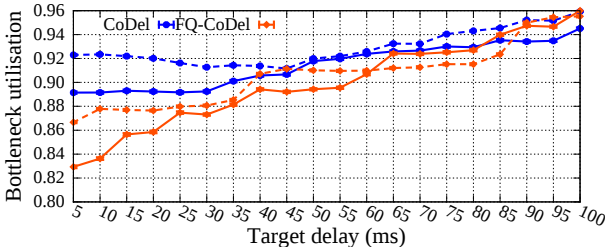


(a) Bottleneck utilisation.

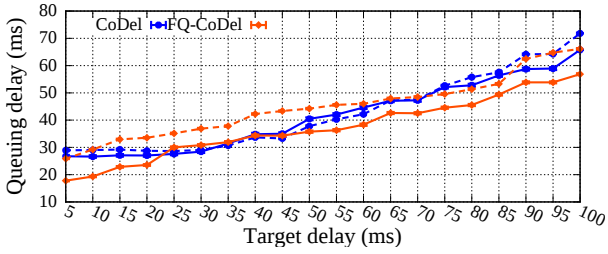


(b) Queuing delay.

Figure 3: Utilisation and delay for various λ, with $\tau = 5$ ms (solid lines: low load; dashed lines: high load).



(a) Bottleneck utilisation



(b) Queuing delay

Figure 4: Utilisation and delay for various τ, with $\lambda = 300$ ms (solid lines: low load; dashed lines: high load).

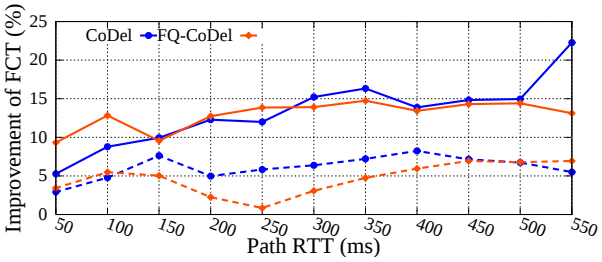
D. Rural and Default parameterisations

We propose to set τ to 65 ms and λ to 300 ms for both CoDel and FQ-CoDel, so that at least 90 % the bottleneck capacity of the constrained network is utilised and the queuing delay limited to a reasonable value, much smaller than the base RTT. While these parameters may be appropriate for other capacity-limited cases, we do not claim they suit every

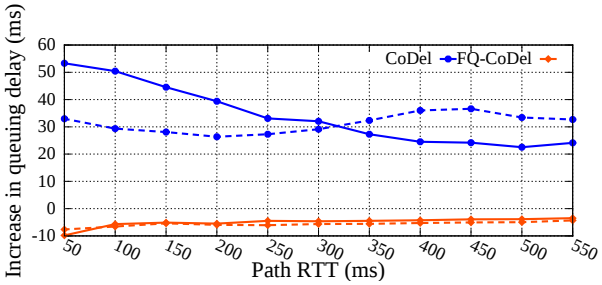
capacity-limited network. We therefore evaluate if this parameterisation is robust to a range of RTT values. We refer to the AQM parameter settings optimised to suit the low-bandwidth, high-RTT scenario studied above as *Rural* parameterisation, in contrast with the *Default* parameterisation of an AQM.

E. RTT sensitivity

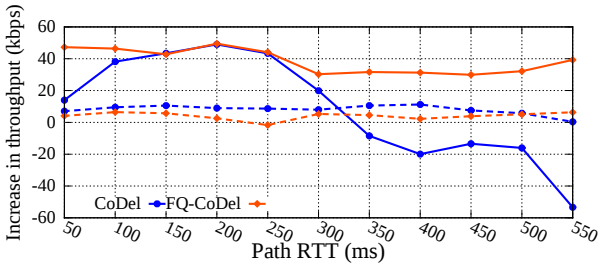
As deployments will experience various RTTs, we compare *Rural* and *Default* parameterisations with a 1 Mbps link across a range of end-to-end RTTs from 50 ms to 550 ms. The improvement imp_M for a given metric M is computed as $imp_M = (M_{Default} - M_{Rural})/M_{Default}$ (for FCT in Fig. 5a) and the increase inc_N for a metric N is computed as $inc_N = N_{Rural} - N_{Default}$ (for queuing delay in Fig. 5b and the bulk throughput in Fig. 5c).



(a) Repeated download of medium-sized files.



(b) Thin-stream flow.



(c) Bulk transfer flow.

Figure 5: Impact of *Rural* parameterisation for various base RTTs (solid lines: low load; dash lines: high load).

Figure 5a shows that *Rural* parameterisation improves the FCT for every RTT. In Fig. 5b, we can see that the queuing delay for thin streams is increased with CoDel with *Rural* parameterisation, because $\tau_{Rural} > \tau_{Default}$, but this is not the case with FQ-CoDel, due to its prioritisation scheme. Except when the RTT is higher than 350 ms and CoDel is used, *Rural* parameterisation increases the bulk throughput.

F. Flows with different RTTs

We next compare the performance of *Rural* and *Default* parameterisations by monitoring a repeated-download flow competing with another similar, "background" flow but having a different base RTT. The RTT ratios of the two flows were varied between 0.09 and 11, i.e., RTT $\in [50, 550]$ ms.

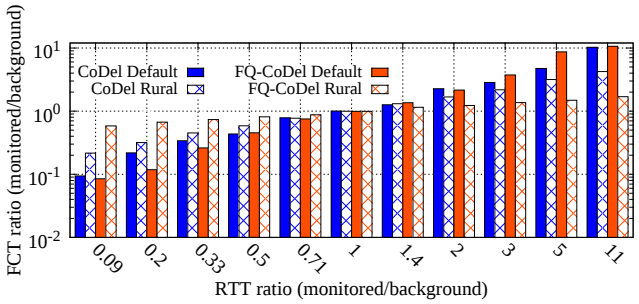


Figure 6: FCT with two competing RTTs.

The results presented in Fig. 6 show that both AQMs with *Rural* parameterisation reduce the FCT for the large RTT flow and reduces the difference of download time for flows that experience different base RTT: the FCT ratio ($\frac{FCT\ Monitored}{FCT\ Background}$) moves closer to one with *Rural* parameterisation, for both small and large RTT ratios.

G. Discussion

We do not claim that AQM would provide the best quality of service for various applications when the link capacity or traffic patterns are different; however, we show that (1) adequately tuned AQM schemes can improve the FCTs and the link utilisation , and that (2) CoDel can be tuned to achieve various trade-offs.

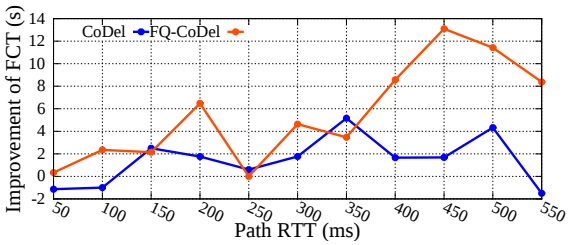
V. Experimental evaluation

A. Performance over a delay-emulated test bed

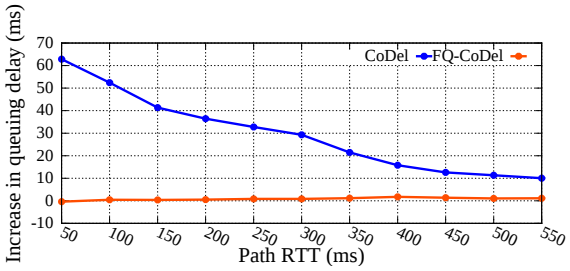
The experiments described in Section III-B were repeated with *Rural* parameterisation and the results compared with those presented in that section. Figure 7 presents the improvement in terms of FCT and the increase in queuing delay and throughput over *Default* parameterisation as explained in § IV-E. ICMP ping flows were generated to measure the queuing delay, shown in Fig. 7b. Figure 7a shows that *Rural* parameterisation improves the FCT for most RTT values. As expected, the higher throughput for the bulk flow, presented in Fig. 7c, results in a higher queuing delay. We notice a difference between the results presented in Figures 5 and 7, which is due to the different traffic characteristics and different source codes of FQ-CoDel in ns-2 and Linux.

B. Performance over a satellite network

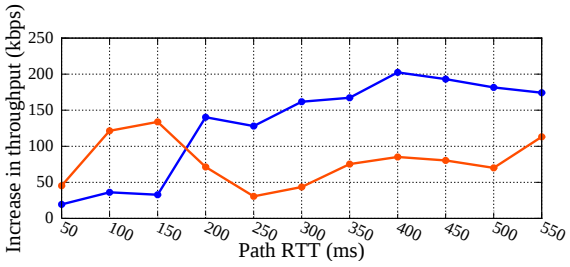
Finally, we present results obtained with a operational rural satellite broadband service: the forward satellite link is between snd_1 and R_1 in Fig. 1, the bottleneck is restricted to 1 Mbps and AQM is used at R_1; in addition to the queue management schemes tested so far, this link allowed also the



(a) Repeated download of medium-sized files.



(b) Thin-stream flow.



(c) Bulk transfer.

Figure 7: Base RTT and *Rural* and *Default* parameterisations

use of Stochastic FQ (SFQ). Figure 8 shows that introducing CoDel or FQ-CoDel with *Default* parameterisation results in (1) a higher FCT than DropTail or SFQ and (2) a lower RTT. With CoDel or FQ-CoDel with *Rural* parameterisation, the RTT is lower than with the other evaluated schemes, without negative impact on the FCT. With adequately parameterised AQM, as opposed to with DropTail, the latency is reduced by $\approx 60\,\%$ (i.e., by 300 ms), which would seriously increase rural broadband users' experience.

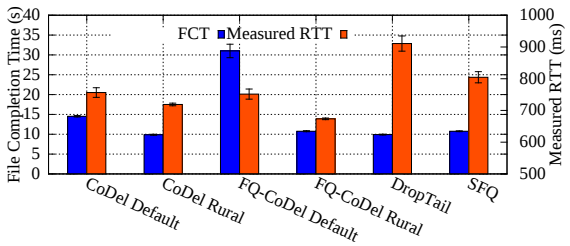


Figure 8: FCT and measured RTT over a satellite network

VI. Conclusion

This paper characterised the current default parameterisation of CoDel and FQ-CoDel within a limited-capacity and large-RTT scenario, which is representative of a rural broadband access network. We show that the default parameters do not offer the desired control of queuing for a single-queue AQM, as claimed in [3]. Consistent with other analysis of PIE, we have proven CoDel can also be tuned to better match the needs of a "non-default" network scenario.

We propose a configuration with a larger target delay and update interval for CoDel: this resulted in reduced download time of medium-sized files, higher bottleneck utilisation and limited queuing delay to an acceptable level, for different traffic profiles. However, this also resulted in queuing delay larger than the *target delay*. Under a less-controlled queuing delay, the tested flow isolation technique, FQ-CoDel, with our parameterisation, provided low latency and was seen to significantly improve performance for delay-sensitive traffic. FQ-CoDel also features priority queuing which improves the performance of applications transmitting a low amount of data. However, we note that in some cases, flow separation may be difficult (e.g., when encryption is used) and CoDel can then control delay to a reasonable level. With our parameterisation, CoDel and FQ-CoDel would allow a better quality of service for users browsing the web from a rural location since the latency is reduce and the capacitiy exploited close to the one without AQM schemes. As a future work, we will evaluate the benefits of our parameterisation for real-time services.

Acknowledgment

This research was supported by the RCUK DE award to the dot.rural Digital Economy Hub; EP/G066051/1 and by the European Community under its Seventh Framework Programme through the Reducing Internet Transport Latency (RITE) project (ICT-317700).

References

[1] J. Gettys, "Bufferbloat: Dark buffers in the Internet," *IEEE Internet Computing*, vol. 15, no. 3, pp. 96–96, 2011.

[2] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *Networking, IEEE/ACM Transactions on*, 1993.

[3] K. Nichols and V. Jacobson, "Controlling queue delay," *ACM Queue*, 2012.

[4] R. Pan, P. Natarajan, C. Piglione, M. Prabhu, V. Subramanian, F. Baker, and B. VerSteeg, "PIE: A lightweight control scheme to address the Bufferbloat problem," in *IEEE HPSR 2013*, 2013, pp. 148–155.

[5] T. Hoeiland-Joergensen, P. McKenney, D. Taht, J. Gettys, and E. Dumazet, "Flowqueue-codel (work in progress)," 2013, IETF.

[6] SamKnows - Commission for Rural Communities, "Mind the Gap: Digital England – A Rural Perspective," June 2014.

[7] L. Townsend, A. Sathiaseelan, G. Fairhurst, and C. Wallace, "Enhanced broadband access as a solution to the social and economic problems of the rural digital divide," *Local Economy*, 2013.

[8] I. Järvinen and M. Kojo, "Evaluating CoDel, PIE, and HRED AQM Techniques with Load Transients," *LCN*, 2014.

[9] J. Schwardmann, D. Wagner, and M. Kühlewind, "Evaluation of ARED, CoDel and PIE," *20th EUNICE*, 2014.

[10] R. Pan, P. Natarajan, C. Piglione, M. S. Prabhu, V. Subramanian, F. Baker, and B. VerSteeg, "PIE: A lightweight control scheme to address the bufferbloat problem. Further Studies– PIE for Data Centers," in *IETF 86 - ICCRG presentation*, 2013.

[11] G. White and R. Pan, "A PIE-Based AQM for DOCSIS Cable Modems (work in progress)," 2014, IETF.

[12] K. Nichols, V. Jacobson, A. McGregor, and J. Iyengar, "Controlled delay active queue management (work in progress)," 2014, IETF.